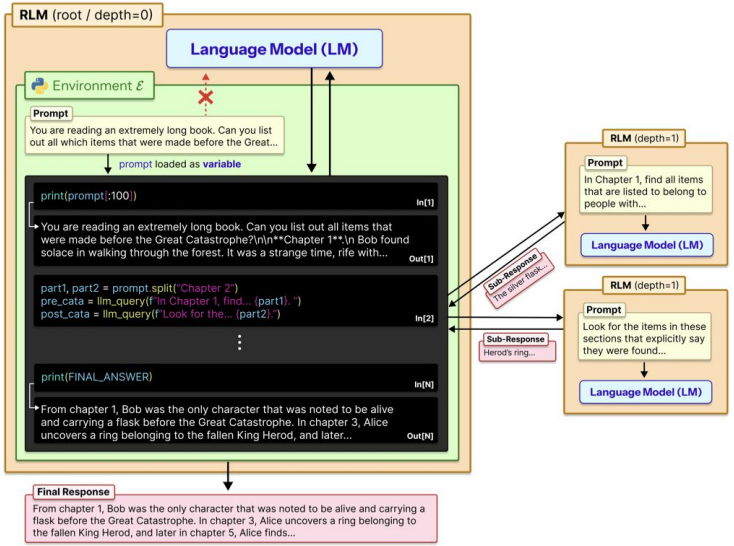


Recursive Language Models

Alex L. Zhang - 1/30/2026

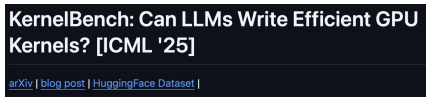
+ Tim Kraska, Omar Khattab

A little about me...

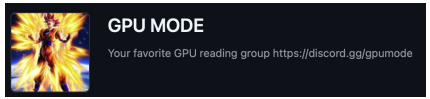
Alex Zhang, winner of the Phillip Goldman '86 Senior Prize. Photo by Lori M. Nichols.

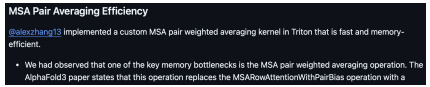
~1.5 year ago

Benchmarks



GPU MODE + BioML





PhD student as of ~5 months ago

Theme: “Recursion” around LMs.

Less is More: Recursive Reasoning with Tiny Networks

Alexia Jolicoeur-Martineau
Samsung SAIL Montréal
alexia.j@samsung.com

Abstract

Hierarchical Reasoning Model (HRM) is a novel approach using two small neural networks recursing at different frequencies. This biologically inspired method beats Large Language models (LLMs) on hard puzzle tasks such as Sudoku, Maze, and ARC-AGI while trained with small models (27M parameters) on small data (~ 1000 examples). HRM holds great promise for solving hard problems with small networks, but it is not yet well understood and may be suboptimal. We propose Tiny Recursive Model (TRM), a much simpler recursive reasoning approach that achieves significantly higher generalization than HRM, while using a single tiny network with only 2 layers. With only 7M parameters, TRM obtains 45% test-accuracy on ARC-AGI-1 and 8% on ARC-AGI-2, higher than most LLMs (e.g., Deepseek R1, o3-mini, Gemini 2.5 Pro) with less than 0.01% of the parameters.

1. Introduction

While powerful, Large Language models (LLMs) can struggle on hard question-answer problems. Given that they generate their answer auto-regressively, there is a high risk of error since a single incorrect token can render an answer invalid. To improve their reliability, LLMs rely on Chain-of-thoughts (CoT) (Wei et al., 2022) and Test-Time Compute (TTC) (Snell et al., 2024). CoTs seek to emulate human reasoning by having the LLM to sample step-by-step reasoning traces prior to giving their answer. Doing so can improve accuracy, but CoT is expensive, requires high-quality reasoning data (which may not be available), and can be brittle since the generated reasoning may be wrong. To further improve reliability, test-time compute can be used by reporting the most common answer out of K or the highest-reward answer (Snell et al., 2024).

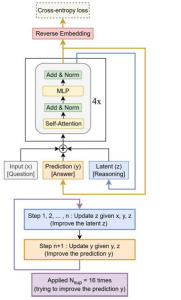


Figure 1. Tiny Recursion Model (TRM) recursively improves its predicted answer y with a tiny network. It starts with the embedded input question x and initial embedded answer y, and latent z. For up to $N_{sup} = 16$ improvements steps, it tries to improve its answer y. It does so by i) recursively updating n times its latent z given the question x, current answer y, and current latent z (recursive reasoning), and then ii) updating its answer y given the current answer y and current latent z. This recursive process allows the model to progressively improve its answer (potentially addressing any errors from its previous answer) in an extremely parameter-efficient manner while minimizing overfitting.

Andy Konwinski About Me

Recursive LLM Prompts

Mar 20, 2023

(The following is copied from the readme at github.com/andyk/recursive_llm); Last updated: April 3, 2023

The idea here is to implement recursion using English as the programming language and an LLM (e.g., GPT-3.5) as the runtime.

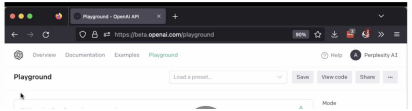
Basically we come up with an LLM prompt which causes the model to return another slightly updated prompt. More specifically, the prompts contain state and each recursively generated prompt updates that state to be closer to an end goal (i.e., a base case).

It's kind of like recursion in code, but instead of having a function that calls itself with a different set of arguments, there is a prompt that returns itself with specific parts updated to reflect the new arguments.

For simplicity, let's start without a base case; here is an infinitely recursive fibonacci prompt:

> *You are a recursive function. Instead of being written in a programming language, you are written in English. You have variables FIB_INDEX = 2, MINUS_TWO = 0, MINUS_ONE = 1, CURR_VALUE = 1. Output this paragraph but with updated variables to compute the next step of the Fibbonaci sequence.*

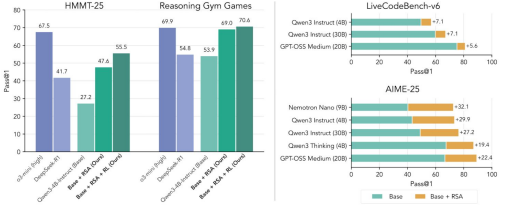
To “run this program” we can paste it into OpenAI playground, and click run, and then take the result and run that, etc.



Recursive Self-Aggregation Unlocks Deep Thinking in Large Language Models

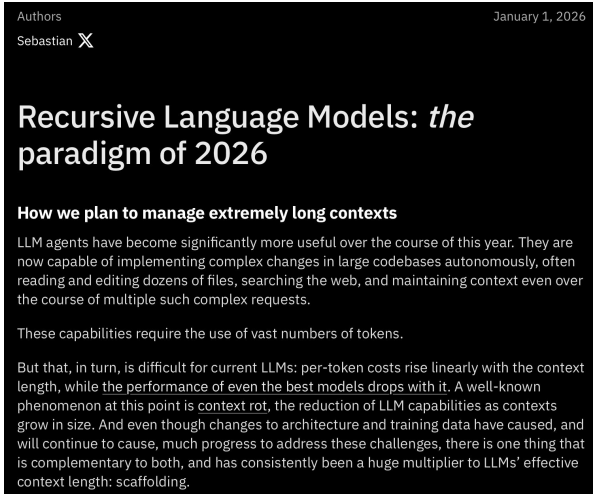
Siddarth Venkatraman[*1,2] Vineet Jain[*1,3] Sarthak Mittal[*1,2] Vedant Shah[1,2]
Johan Obando-Ceron[1,2] Yoshua Bengio[1,2,4,7] Brian Bartoldson[5] Bhavya Kailkhura[5]
Guillaume Lajoie[1,2,7] Glen Berseth[1,2,7] Nikolay Malkin[6,8] Moksh Jain[1,2]
[*]Equal contribution [1]Mila - Québec AI Institute [1]Université de Montréal [3]McGill University
[4]LawZero [5]LLNL [6]University of Edinburgh [7]CIFAR AI Chair [8]CIFAR Fellow

Paper Code arXiv (coming soon)



Recursive Self-Aggregation (RSA) substantially improves Pass@1 across tasks and model architectures. RSA enables the much smaller Qwen3-4B-Instruct-2507 to match the performance of larger reasoning models such as DeepSeek-R1 and o3-mini (high). These gains are further amplified through our proposed aggregation-aware RL framework.

Start of 2026: Context management is key



Authors

Sebastian

January 1, 2026

Recursive Language Models: *the* paradigm of 2026

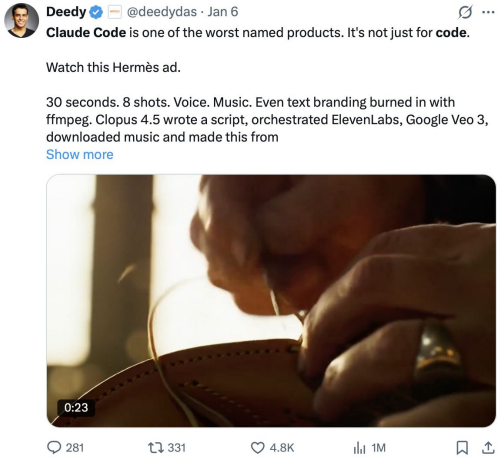
How we plan to manage extremely long contexts

LLM agents have become significantly more useful over the course of this year. They are now capable of implementing complex changes in large codebases autonomously, often reading and editing dozens of files, searching the web, and maintaining context even over the course of multiple such complex requests.

These capabilities require the use of vast numbers of tokens.

But that, in turn, is difficult for current LLMs: per-token costs rise linearly with the context length, while the performance of even the best models drops with it. A well-known phenomenon at this point is context rot, the reduction of LLM capabilities as contexts grow in size. And even though changes to architecture and training data have caused, and will continue to cause, much progress to address these challenges, there is one thing that is complementary to both, and has consistently been a huge multiplier to LLMs' effective context length: scaffolding.

Jan 1. Prime Intellect reveals they are working on RLMs.



Deedy @deedydas · Jan 6

Claude Code is one of the worst named products. It's not just for **code**.

Watch this Hermès ad.

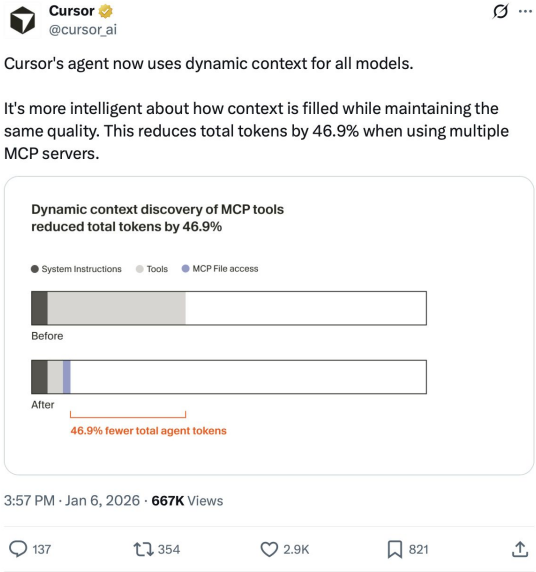
30 seconds. 8 shots. Voice. Music. Even text branding burned in with ffmpeg. Clopus 4.5 wrote a script, orchestrated ElevenLabs, Google Veo 3, downloaded music and made this from

Show more

0:23

281 331 4.8K 1M

Past few months: people are trying Claude Code for non-coding tasks.



Cursor

@cursor_ai

Cursor's agent now uses dynamic context for all models.

It's more intelligent about how context is filled while maintaining the same quality. This reduces total tokens by 46.9% when using multiple MCP servers.

3:57 PM · Jan 6, 2026 · **667K** Views

137 354 2.9K 821

Jan 6. Cursor release dynamic context.

Long context + Language Models is unsatisfactory...

Why context engineering is important to building capable agents

Despite their speed and ability to manage larger and larger volumes of data, we've observed that LLMs, like humans, lose focus or experience confusion at a certain point. Studies on needle-in-a-haystack style benchmarking have uncovered the concept of context rot: as the number of tokens in the context window increases, the model's ability to accurately recall information from that context decreases.

I asked my LM to finish carving the pumpkin joke it started yesterday. It said, "Pumpkin? What pumpkin?" — the context completely rotted.

Long context + Language Models is unsatisfactory...

Why context engineering is important to building capable agents

Despite their speed and ability to manage larger and larger volumes of data, we've observed that LLMs, like humans, lose focus or experience confusion at a certain point. Studies on needle-in-a-haystack style benchmarking have uncovered the concept of context rot: as the number of tokens in the context window increases, the model's ability to accurately recall information from that context decreases.

I asked my LM to finish carving the pumpkin joke it started yesterday. It said, "Pumpkin? What pumpkin?" — the context completely rotted.

kwindla @kwindla

For the things I do (voice agents / realtime multi-turn / multi-step function calling harnesses) needle in a haystack benchmarks don't map at all to real-world long context performance.

I care almost entirely about
- Can you combine your system instructions with the last 30 turns or so of information to "understand the assignment every turn?"
- Can you reliably call functions (with the correct arguments) deep into a multi-turn conversation.

Generally, for non-trivial instruction complexity, today's models are not great in multi-turn contexts compared to "normal" use.

So ... context engineering, amiright @dexhorthy?

> **Nicolay Gerold** @nicolaygerold · Aug 31
> After 300k tokens gpt5's tool calling seems to be pretty much gone. Starts to write python scripts because it fails repetitive edit file calls.

3:07 PM · Aug 31, 2025 · **4,208** Views

Some **intuition** on Recursive Language Models (RLMs)

1. Handling long distributions of data is “long-tailed” – i.e., it rarely occurs in practice.

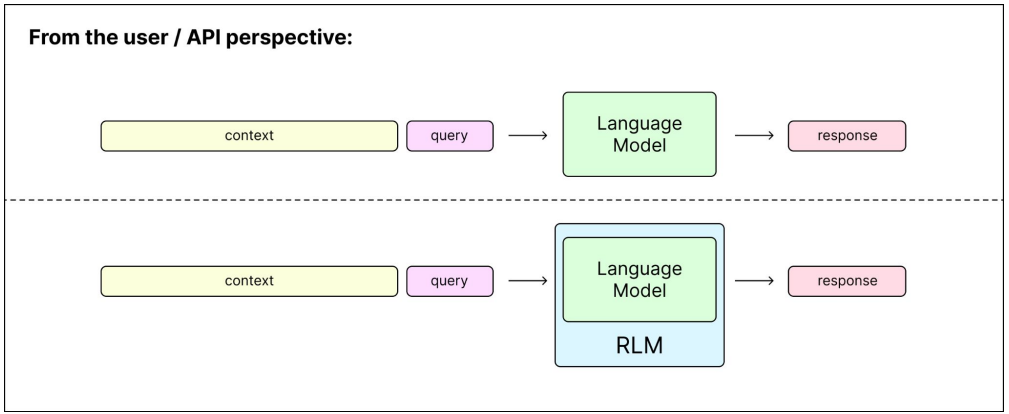
2. Natively supporting longer contexts in a base LM is extremely expensive (requires re-training + scaling a new model).

3. We want the **illusion** of an LM that can handle near-infinite context without degradation, which works within the current limitations of LMs.

Bitter Lesson view of Recursive Language Models (RLMs)

We treat LM calls as a blackbox, and design human-engineered scaffolds around them.

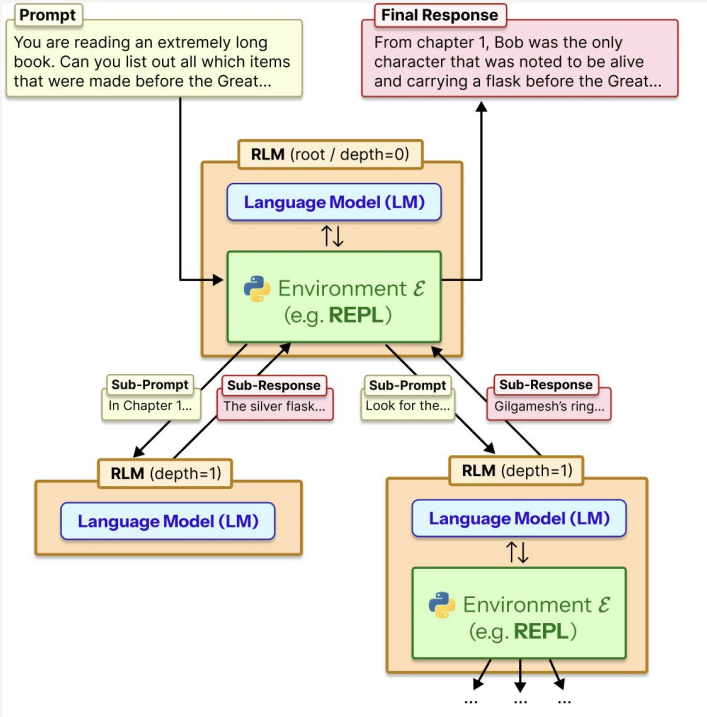
Q: Why don't we defer the management of input context, subcalls, and outputs to the language model? Then directly optimize this functionality (e.g. with RL)?



Recursive Language Models

Recursive Language Models (RLMs) are a general-purpose inference paradigm for dramatically scaling the effective input and output lengths of modern LLMs.

The **key insight** is that long prompts should not be fed into the neural network (e.g., Transformer) directly but should instead be treated as part of the environment that the LLM can symbolically interact with.



The most important slide (1/2)

What are the essential components of an RLM? What has been missing?

- A **symbolic handle** to the prompt P, which offloads potentially long inputs from directly being ingested by a single language model call.
- **Persistent symbolic programming**, which allows the language model M to interact with and answer the prompt through intermediate stateful variables.
- The ability to perform **symbolic recursion**, which allows the model to invoke itself using programmatic patterns to decompose and solve problems.

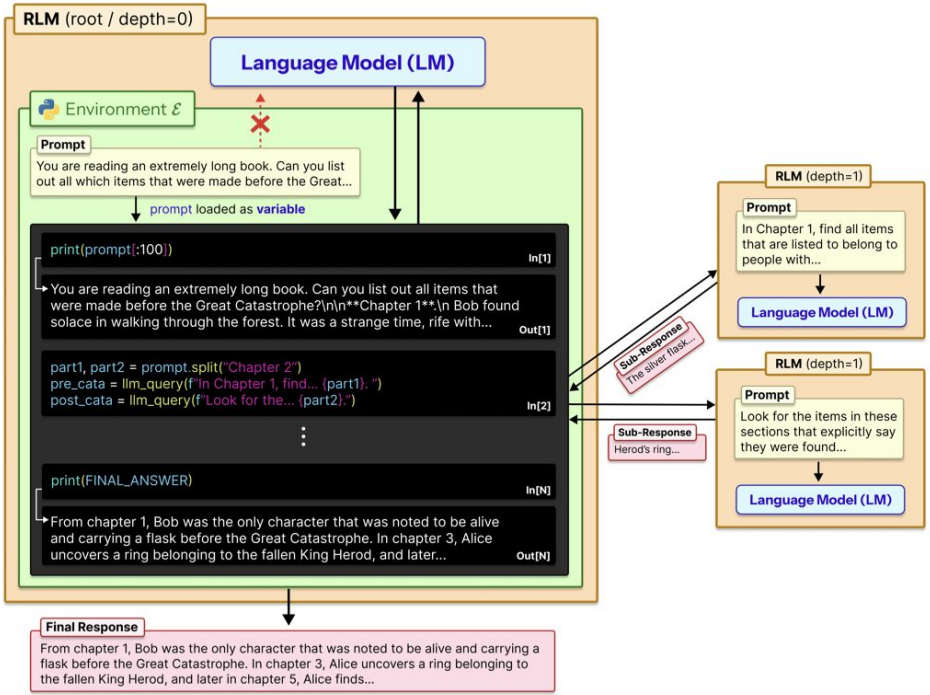
The most important slide (2/2)

“The ability to perform **symbolic recursion**, which allows the model to invoke itself using programmatic patterns to decompose and solve problems.”

Does the sub-agent live **inside** or **outside** the REPL? Are these representations equal?

- What do context folding, CC, Codex, Terminus-Bench sub-agents, and MOST agent implementations implement incorrectly? The sub-agent is a tool outside of the REPL – it is **not** programmatic.

RLMs reason and delegate through a REPL



Algorithm 1 Recursive Language Model (RLM) Call

Input: Input prompt P, maximum turns $T_{\max}$
Output: Final output O

LOAD PROMPT AND SETUP REPL.
turns $\leftarrow$ [ ]
state $\leftarrow$ {"prompt": P}
tools $\leftarrow$ [$\texttt{sub\_RLM}_{\mathcal{M}}()$, ...]

for $t = 1$ **to** $T_{\max}$ **do**
  (reasoning, code) $\leftarrow$ $\texttt{LLM}_{\mathcal{M}}$(turns)
  state $\leftarrow$ $\texttt{EXEC\_REPL}_{\mathcal{E}}$(state, tools, code)
  stdout $\leftarrow$ format_str(state)
  turns $\leftarrow$ turns.append({reasoning, code, stdout})
  if *HasFinalOutput(state)* **then**
    return ExtractFinalAnswer(state)
  end
end
return $\texttt{LLM}_{\mathcal{M}}(turns)$

Experiments (+ what we wanted to show)

1. RLMs deal with **context rot** by avoiding base LM calls over huge contexts.

2. RLMs natively handle “**near infinite**” context. i.e. just feed in whatever you want into the model and let it deal with it.

3. RLMs scale with strong performance on long tasks with **dense** information.

Tasks (1/2); sparse

S-NIAH: Single needle in the haystack, find a hidden message in a huge sequence of text.

BrowseComp+ (1k) (Chen et al., 2025). Multi-hop DeepResearch benchmark to find answer to complex queries given 1K document corpus.

LongBench-v2 CodeQA (Bai et al., 2024). Multiple choice QA over large code repositories.

Tasks (2/2); dense

Setup. The `trec_coarse` split consists of 6 different types of queries to answer distributional queries about a giant list of "question" entries. For example, one question looks like:

```
For the following question, only consider the subset of instances that are associated with user IDs 67144, 53321, 38876, 59219, 18145, 64957, 32617, 55177, 91019, 53985, 84171, 82372, 12053, 33813, 82982, 25063, 41219, 90374, 83707, 59594. Among instances associated with these users, how many data points should be classified as label 'entity'? Give your final answer in the form 'Answer: number'.
```

The query is followed by ~3000 - 6000 rows of entries with associated user IDs (not necessarily unique) and instances that **are not explicitly labeled** (i.e. the model has to infer the labeling to answer). They look something like this:

```
Date: Dec 12, 2022 || User: 63685 || Instance: How many years old is Benny Carter ?
Date: Dec 30, 2024 || User: 35875 || Instance: What war saw battles at Parrot 's Beak and Black Virgin ?
Date: Apr 13, 2024 || User: 80726 || Instance: What Metropolis landmark was first introduced in the Supe
Date: Feb 29, 2024 || User: 59320 || Instance: When was Calypso music invented?
...
```

OOLONG (Bertsch, 2025).

Task 1

In the above data, list all pairs of user IDs (no duplicate pairs, list lower ID first) where both users have at least one instance with a numeric value or location. Each of the questions can be labelled as one of the labels (the data does not provide the labels, you need to figure out the label from the semantics of the question): description and abstract concept, entity, human being, numeric value, location, abbreviation. In your answer, list all pairs in the format (user_id_1, user_id_2), separated by newlines.

OOLONG-Pairs

Main Experiments

Table 1: Performance comparison of different methods across long-context benchmarks of varying complexity. In gray is the average API cost ± the standard deviation of each method on each task. * indicates runs where the method ran into input context limits.

Model	CodeQA	BrowseComp+ (1K)	OOLONG	OOLONG-Pairs
Task Length N (tokens)	23K-4.2M	6M-11M	131K	32K
Qwen3-Coder-480B				
Base Model	20.00* ($0.13 ± $0.08)	0.00* (N/A) ± (N/A)	36.00 ($0.06 ± $0.00)	0.06 ($0.05 ± $0.01)
CodeAct (+ BM25)	24.00* ($0.17 ± $0.08)	12.66 ($0.39 ± $0.50)	38.00 ($1.51 ± $1.09)	0.28 ($1.54 ± $0.35)
Summary agent	50.00 ($1.26 ± $1.50)	38.00 ($8.98 ± $2.12)	44.06 ($0.15 ± $0.01)	0.31 ($0.05 ± $0.00)
RLM	56.00 ($0.92 ± $1.23)	44.66 ($0.84 ± $0.63)	**48.00** ($0.61 ± $0.49)	**23.11** ($1.02 ± $0.52)
RLM (no sub-calls)	**66.00** ($0.18 ± $0.58)	**46.00** ($0.82 ± $0.69)	43.50 ($0.32 ± $0.13)	17.34 ($1.77 ± $1.23)
GPT-5				
Base Model	24.00* ($0.13 ± $0.07)	0.00* (N/A) ± (N/A)	44.00 ($0.14 ± $0.02)	0.04 ($0.16 ± $0.10)
CodeAct (+ BM25)	22.00* ($0.06 ± $0.08)	51.00 ($0.71 ± $1.20)	38.00 ($0.61 ± $1.06)	24.67 ($0.75 ± $0.43)
Summary agent	58.00 ($1.31 ± $1.46)	70.47 ($0.57 ± $0.10)	46.00 ($0.13 ± $0.01)	0.01 ($0.13 ± $0.09)
RLM	**62.00** ($0.11 ± $0.10)	**91.33** ($0.99 ± $1.22)	**56.50** ($0.43 ± $0.85)	**58.00** ($0.33 ± $0.20)
RLM (no sub-calls)	58.00 ($0.18 ± $0.56)	88.00 ($0.44 ± $0.90)	36.00 ($0.37 ± $0.42)	43.93 ($0.69 ± $1.16)

Inference Costs are Long-Tailed

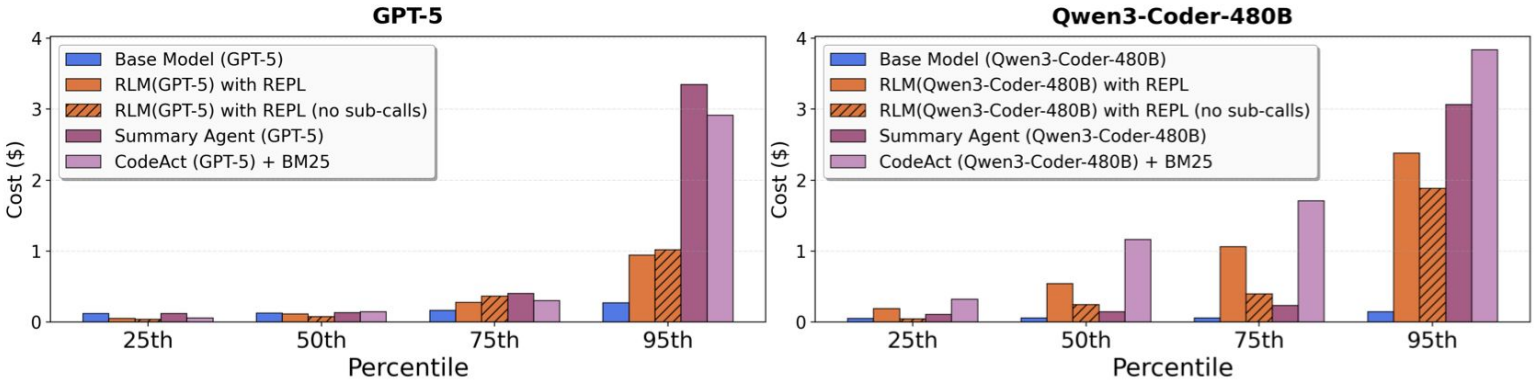


Figure 3: Cost of RLM and baselines described in §2.2 plotted at the 25th, 50th, 75th, and 95th percentile of total API cost. We observe comparable or even lower costs for RLMs at the 50th percentile, but sharp increases at the tail end due to potentially long RLM trajectories.

How does RLM vs. LM performance scale?

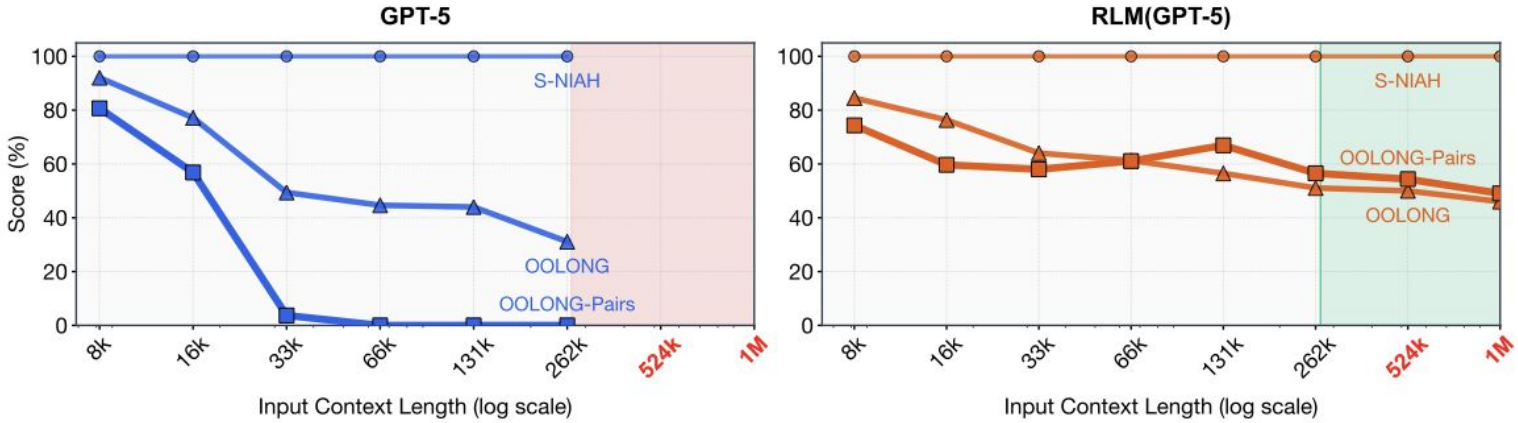
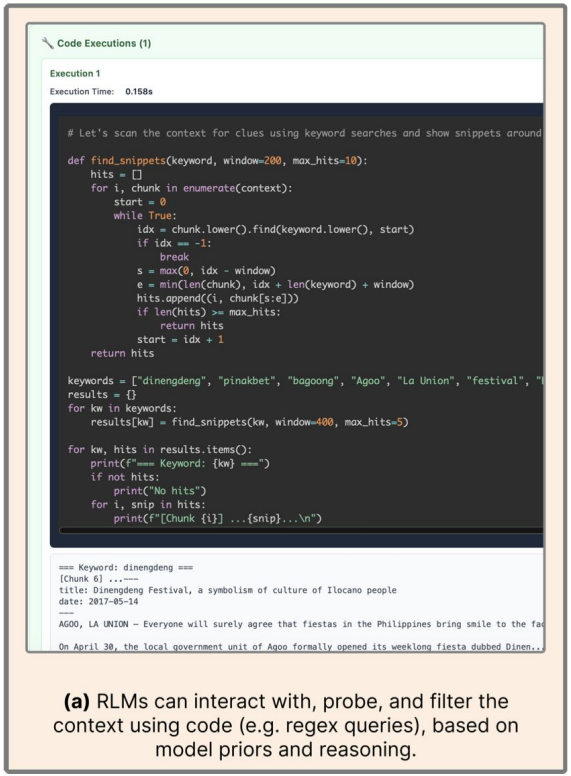
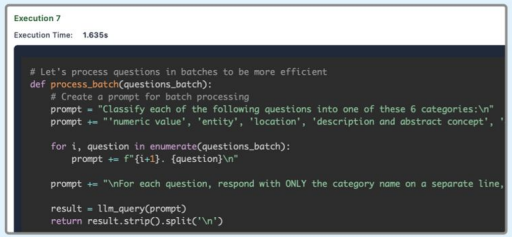


Figure 1: A comparison of GPT-5 and a corresponding RLM on three long-context tasks of increasing complexity: **S-NIAH**, **OOLONG**, and **OOLONG-Pairs**. For each task, we scale the input length from 2^{13} to 2^{18}. GPT-5 performance degrades significantly as a function of both input length and task complexity, while the RLM maintains strong performance. Inputs beyond the red region do not fit in GPT-5's context window of 272K tokens, but the RLM handles them effectively. Additional experiments across other models, methods, and benchmarks are in §2.

General RLM Behavior



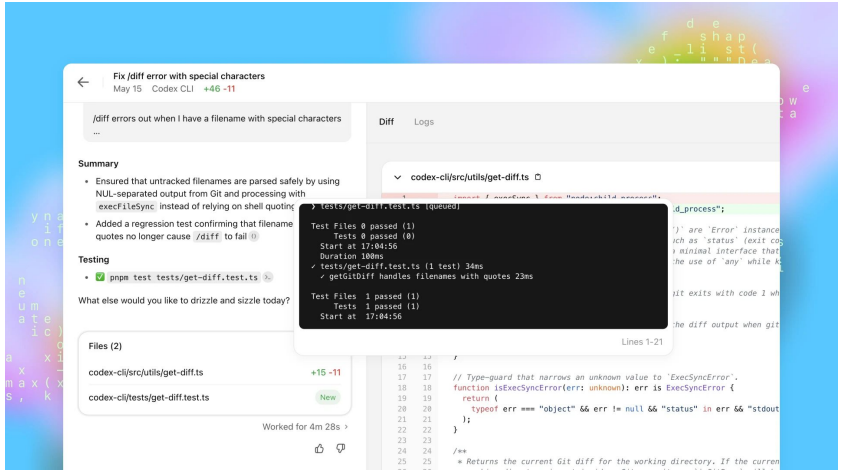
(a) RLMs can interact with, probe, and filter the context using code (e.g. regex queries), based on model priors and reasoning.



(b) RLMs defer reasoning over its large context by querying recursive LM calls.

(c) RLMs can stitch recursive LM outputs to form longer, composite outputs.

How do RLMs relate to **coding scaffolds**?



Future Directions

1. **Training**. How do we train LMs to be more effective as RLMs; either bootstrapping with RLVR or distilling into a small model?

2. **Inference.** How can we make RLM inference faster and more efficient?

3. **Domains.** Which settings are RLMs most useful? Can we use them for long-horizon tasks (e.g. agentic, evolutionary methods, etc.)